UNIVERSITY
OF TRENTO

DEPARTMENT OF INFORMATION AND COMMUNICATION TECHNOLOGY

38050 Povo – Trento (Italy), Via Sommarive 14
http://www.dit.unitn.it

COMPUTING THE ACCESSIBILITY RELATION FOR THE AMBIENT CALCULUS

Radu Mardare and Corrado Priami

October 2003

Technical Report # DIT-03-054

Computing the accessibility relation for the Ambient Calculus*

Radu Mardare and Corrado Priami
University of Trento

Abstract

We present some algorithms to compute the reductions of Ambient Calculus and to describe mobility and dynamic hierarchies of locations. The main idea is to treat each ambient program as a complex of a set-theoretical structure for describing the hierarchy of locations, a function for naming the nodes of the structure, and a function for registering the capabilities of each node. These complexes, named state-processes, can be seen as states for a propositional branching tree logic built upon the Ambient Calculus. We develop here, the algorithms for implementing the accessibility relation between the states of this logic. Our algorithms compute the evolution of the truth values for the atomical propositions during the firing of capabilities. The model presented in this paper permits to re-use algorithms for model checking temporal logic to approach the Ambient Calculus.

1 Introduction

The Ambient Calculus [4] is a useful tool to study mobility when processes may reside within a hierarchy of locations. For describing properties of mobile computations as well as the hierarchy of locations and modifications of this hierarchy in time it is necessary to develop a logic strongly based on Ambient Calculus. Ambient Logic [3, 2] satisfies these requirements using two modalities, one for spatial assertions and an other for temporal ones. Combining these two is possible to obtain a quite expressive analysis of mobile computation. We developed, in a companion paper [6], an alternative to this logic - a CTL* one [5] - able to satisfy the same requirements.

There are many advantages in working with a temporal logic. First, the possibility of having path assertions together with state ones that allow to express properties difficult to express with other modal logics [5]. Second, the possibility of having model checking for our calculus reusing some software already developed for temporal logics such as SMV, NuSMV, SiMpLer, VIS.

*Work partially supported by the IST-FET project DEGAS and the MIUR-COFIN01 project MEFISTO.

The requirements of a propositional temporal logic model checker is to describe the possible states of the system, to assign truth values to atomical propositions for each state, and to describe the accessibility relation between states. Fulfilling these requirements will be the subject of this paper.

We construct a model for Ambient Calculus that help us interpreting states. We associate to each ambient calculus program a state-process, a set-theoretical entity consisting of

1. a flat system of equations [1] that describes the hierarchical structure of our program
2. a function that associates with each node of the structure a name of ambient or atomical process
3. a function that associates to each node a list of capabilities

In [6] we defined over state-processes a bisimulation relation and we proved that this can depict the structural congruence over ambient processes. In this way, the state-processes can speak for ambient processes up to the bisimulation level.

Further we will use the state-processes as states for our temporal logic.

2 State-Processes

In this section we will briefly recall the definition of state-processes and the construction of a state-process for a given ambient process, for more see [6]. We work inside Zermelo-Fraenkel system of Set Theory ZFC [1] with the Foundation Axiom (FA). We assume a class $\mho$ of urelements, set-theoretical entities which are not sets (they do not have elements) but can be elements of sets. The urelements together with the empty set $\emptyset$ will generate all the sets we will work with (sometimes sets of sets).

Definition 2.1. A *flat system of equations* is a triple $\mathcal{E} = \langle X, A, e\rangle$ with $X \subseteq \mho$ and A, sets such that $X \cap A = \emptyset$, and a function $e : X \to \mathcal{P}(X \cup A)$. X is called the set of *indeterminates* of $\mathcal{E}$, A is called the set of *atoms* of $\mathcal{E}$. For each $v \in X$, the set $b_v \stackrel{def}{=} e_v \cap X$ is called the set of indeterminates on which v immediately depends. Similarly, the set $c_v \stackrel{def}{=} e_v \cap A$ is called the set of atoms on which v immediately depends (we wrote e_v for $e(v)$). A *solution* to $\mathcal{E}$ is a function s with domain X satisfying $s_x = \{s_y | y \in b_x\} \cup c_x$ for each $x \in X$. The *solution-set* of a flat system $\mathcal{E}$ of equations is the set $ss(\mathcal{E}) = \{s_v | v \in X\}$.

Definition 2.2. A *state-process* is a triple $S = \langle \mathcal{E}, f, F\rangle$ where:

- $\mathcal{E} = \langle X, A, e\rangle$ is a flat system of equations over A;
- $f : X \cup A \to \mathcal{N}$ is a function that associates to each $v \in X \cup A$ a name from the set $\mathcal{N} = \mathfrak{N} \cup \mathfrak{P} \cup (\mathbb{N} \times \mathbb{N})$, where $\mathfrak{N}$ is the class of ambient names, $\mathfrak{P}$ is the class of atomical process names and $\mathbb{N}$ is the set of natural numbers (the use of $\mathbb{N} \times \mathbb{N}$ as names, will help us to handle private names). f satisfies $f(X) \subseteq \mathfrak{N} \cup (\mathbb{N} \times \mathbb{N})$ and $f(A) \subseteq \mathfrak{P}$;

- $F : X \cup A \to \mathfrak{Str}$, where $\mathfrak{Str} = \{c_1, c_2, c_3, ...\}^*$ with $c_1, c_2, c_3, ... \in \{\epsilon, in\ n, out\ n, input\ n, output \mid n \in \mathfrak{N} \cup (\mathbb{N} \times \mathbb{N})\}$

The intuition behind these definitions is that an ambient calculus program is nothing more than a structure of type *boxes inside boxes* together with an assignment of names (of ambients and unspecified processes) and an assignment of capabilities to each node of the structure. A flat system of equations can completely describe a *boxes inside boxes structure* [1], while the functions f and F preserve the assignments information.

The way in which a flat system can describe a structure is by choosing urelements, first for the first level ambients, then for their children, then for the children of children, and so on, by induction on the structure of the ambient program, until we touch the bottom level of the structure, i.e. an empty ambient or an atomical process. Since ambient programs are finite, the look up of names inside the ambients will finish after a finite number of steps.

For example, consider the ambient calculus program:

$$m\ [\ open\ n.Q \mid s\ [\ out\ m.in\ m.n\ [\ out\ s.open\ s.P \mid R\ ]\ ]\ ] \mid n[P]. \tag{2.1}$$

As a general rule, we embed our program into a *master ambient*[1]. Our program become:

$$u[m[open\ n.Q|s[out\ m.in\ m.n[out\ s.open\ s.P|R]]]|n[P]] \tag{2.2}$$

We choose an urelement $\alpha \in \mathfrak{U}$ and consider $f(\alpha) = u$. u has two children, m and n, both ambients. We choose an urelement $\beta \in \mathfrak{U} \setminus \{\alpha\}$ and then we choose $\gamma \in \mathfrak{U} \setminus \{\alpha, \beta\}$, and we define $f(\beta) = m$, $f(\gamma) = n$. Consider now the children of m. We choose $q \in \mathfrak{U} \setminus \{\alpha, \beta, \gamma\}$, $f(q) = Q$ then $\delta \in \mathfrak{U} \setminus \{\alpha, \beta, \gamma, q\}$ with $f(\delta) = s$. For the child of n we choose $p \in \mathfrak{U} \setminus \{\alpha, \beta, \gamma, q, \delta\}$ and $f(p) = P$. Q is an atomical process so, on this branch, the analysis is over. The ambient s has only one child, an ambient named n, like an other ambient met already. The fact that we already chose an urelement for an ambient with the same name, will not stop us to choose a new urelement. So, we will choose a new $\mu \in \mathfrak{U} \setminus \{\alpha, \beta, \gamma, p, \delta, q\}$ and $f(\mu) = n$. Then, going to the children of this n, we choose $p' \in \mathfrak{U} \setminus \{\alpha, \beta, \gamma, q, \delta, p, \mu\}$ with $f(p') = P$ and $r \in \mathfrak{U} \setminus \{\alpha, \beta, \gamma, p, \delta, q, \mu, p'\}$, $f(r) = R$. And the action of choosing urelements stops here because between the children of the last n there is no ambient. Thus, we can define $f : \{\alpha, \beta, \gamma, p, \delta, q, \mu, p', r\} \to \mathcal{N}$ by: $f(\alpha) = u$, $f(\beta) = m$, $f(\gamma) = n$, $f(\delta) = s$, $f(\mu) = n$, $f(q) = Q$, $f(p) = P$, $f(p') = P$, $f(r) = R$, and $\{u, m, s, n\} \subseteq \mathfrak{N}$, $\{P, Q, R\} \subseteq \mathfrak{P}$.

Note that f is not injective because $f(p) = f(p')$ and $f(\gamma) = f(\mu)$.

We define $X = \{u \in \mathfrak{U} \mid f(u) \in \mathfrak{N} \cup (\mathbb{N} \times \mathbb{N})\}$ and $A = \{u \in \mathfrak{U} \mid f(u) \in \mathfrak{P}\}$, which in our example became: $A = \{p, q, p', r\}$ and $X = \{\alpha, \beta, \gamma, \delta, \mu\}$.

We can define now a flat system of equations $\mathcal{E} = \langle X, A, e\rangle$ with $e : X \to \mathcal{P}(X \cup A)$. For $w \in X$, $e_w \overset{def}{=} \{e_v \mid v \in X,\ v \text{ is a child of } w\} \cup \{p \mid p \in A,\ p \text{ is a child of } w\}$. We have

[1]This is a technical trick that is not disturbing our analysis because of the rule (RedAmb): $P \to Q \Rightarrow n[P] \to n[Q]$, [4]

$e_\alpha = \{e_\beta, e_\gamma\} \cup \emptyset = \{e_\beta, e_\gamma\}$ $\quad e_\beta = \{e_\delta\} \cup \{q\} = \{e_\delta, q\}$ $\quad e_\gamma = \emptyset \cup \{p\} = \{p\}$
$e_\delta = \{e_\mu\} \cup \emptyset = \{e_\mu\}$ $\quad e_\mu = \emptyset \cup \{p', r\} = \{p', r\}$

These are the equations of our system. Now we have to define the function $F : X \cup A \to \mathfrak{Str}$ where $\mathfrak{Str} = \{c_1, c_2, c_3, ...\}^*$ with $c_i \in \{\epsilon, in\ n, out\ n, open\ n, input\ n, output/n \in \mathfrak{N} \cup (\mathbb{N} \times \mathbb{N})$ for which $\exists x \in X$ s.t. $f(x) = n\}$. c_i could take a finite number of values that linearly depends on the number of ambients in our initial ambient calculus formula. F will associate to each $x \in X \cup A$ the string of prefixes found in front of the ambient or atomical process associated to this x by f. Each string, after a finite number of steps, will end up only with null capability ϵ.

In our example $F : X \cup A \to \mathfrak{Str}$
$F(\alpha) = \langle \varepsilon, \varepsilon, ... \rangle$ because there is no prefix in front of $u = f(\alpha)$;
$F(\beta) = \langle \varepsilon, \varepsilon, ... \rangle$ because there is no prefix in front of $m = f(\beta)$;
$F(\gamma) = \langle \varepsilon, \varepsilon, ... \rangle$ because there is no prefix in front of $n = f(\gamma)$ (n child of u);
$F(\delta) = \langle \varepsilon, \varepsilon, ... \rangle$ because there is no prefix in front of $s = f(\delta)$;
$F(\mu) = \langle out\ m, in\ m, \epsilon \rangle$ because there are two prefixes, *out* m and *in* m, in this order, in front of $n = f(\mu)$ (n child of s);
$F(q) = \langle open\ n, \epsilon \rangle$ because there is only one prefix, *open* n, in front of $Q = f(q)$;
$F(p) = \langle \varepsilon, \varepsilon, ... \rangle$ because there is no prefix in front of $P = f(p)$;
$F(p') = \langle out\ s, open\ s, \epsilon \rangle$ because there are two prefixes, *out* s and *open* s, in this order, in front of $P = f(p')$ (the child of the first n);
$F(r) = \langle \varepsilon, \varepsilon, ... \rangle$ because there is no prefix in front of $R = f(r)$.

The prefix *output* will be used to encode the process $\langle M \rangle$. We will treat this process as being $output.M[\ ]$, *output* will act like any other prefix and $M[\ ]$ will be an empty ambient.

Concerning the new name situation, we will accept, as possible names of ambients, ordered pairs of natural numbers. So, for the first new name found in our analysis we will use the name $\langle 1, 1 \rangle$ in place of n[2], for the second new name we will use $\langle 2, 1 \rangle$, for the third $\langle 3, 1 \rangle$ and so on, each new name found will receive $\langle k, 1 \rangle$ meaning that it is the k^{th} new name found. This approach allows to combine our formula with others for which we already constructed the state-process. In this way all the names in the second formula will receive names as $\langle k, 2 \rangle$ meaning that is the k^{th} new name of the second formula, and so on, the k^{th} new name of the l^{th} formula will receive the name $\langle k, l \rangle$.

Consider $\quad open\ n.P \mid (\nu n)(n[Q] \mid m[in\ n.P \mid R]) \mid n[S]$
We treat this program like being:

$open\ n.P \mid \langle 1, 1 \rangle[Q] \mid m[in\ \langle 1, 1 \rangle.P \mid R]) \mid n[S]$, where $\langle 1, 1 \rangle$ is a name of an ambient. This attitude will not change the future of our program.

More details and the description of the relation between ambient processes and state-processes are in [6].

For the sake of presentation we adopt the notation: $\langle c_1, c_2, c_3, ...c_n \rangle - \langle c_1 \rangle = \langle c_2, c_3, ...c_n \rangle$.

[2] We will replace in the ambient calculus formula, all the occurrences of n inside the scope of (νn), being ambients or capabilities, with $\langle 1, 1 \rangle$

3 The Logic

To define a CTL* logic [5] we need a structure $\mathfrak{M} = (S_0, \mathfrak{S}, \mathfrak{R}, \mathfrak{L})$ where S_0 is an initial state of our model, $\mathfrak{S}$ is the class of all possible states in our model, $\mathfrak{R} \subseteq \mathfrak{S} \times \mathfrak{S}$ is the accessibility relation between states, and $\mathfrak{L} : \mathfrak{S} \longrightarrow \mathcal{P}(AP)$ is a function which associates to each state $S \in \mathfrak{S}$ a set of atomical propositions $\mathfrak{L}(S) \subseteq \mathcal{P}(AP)$ - that we interpret as the set of true atomical propositions in the state S (AP will be the class of atomical propositions). All this construction, together with the semantic and the syntax of our logic, is studied in [6]. The purpose of this paper is to focus on describing the accessibility relation between states and to provide some algorithms to calculate this.

Assume that $S_0 = \langle \mathcal{E}_0, f_0, F_0 \rangle$ is our initial state, with $\mathcal{E}_0 = \langle X_0, A_0, e^0 \rangle$. Then we define $\mathfrak{S}$ as the class of all state-processes $S = \langle \mathcal{E}, f, F \rangle$ with $\mathcal{E} = \langle X_0, A_0, e \rangle$. Now we will define $AP = \{x \mathbf{in} y \mid x, y \in X_0 \cup A_0\}$. In our logic $x \mathbf{in} y$ is just an atomical proposition and x, y just letters. The cardinality of AP is $C^2_{card(X_0 \cup A_0)}$ which depends (polynomial) on the number of atomical processes and ambients in the ambient process corresponding to S_0.

Next we define $\mathfrak{L} : \mathfrak{S} \to \mathcal{P}(AP)$ by

$\mathfrak{L}(S) = \{x \mathbf{in} y \mid x \in e_y$ if $x \in A$ or $e_x \in e_y$ if $x \in X\}$

From CTL* semantic we have $\mathfrak{M}, S \models x \mathbf{in} y$ iff $x \mathbf{in} y \in \mathfrak{L}(S)$, as expected.

4 Algorithms for the accessibility relation

The accessibility relation is defined, inductively, on the structure of the initial state, by analyzing all of its possible derivatives. We have to analyze how the use of the existing prefixes of the ambient process will influence the corresponding state-processes. We define an algorithm of evolution of state-process for each reduction rule of the ambient calculus. We use here two matrices that describe the flat system $\mathcal{E}$ and the action of functions F and f, respectively.

Consider the ambient process (2.2) already handled earlier. We have $X = \{\alpha, \beta, \gamma, \delta, \mu\}$, $A = \{q, p, p', r\}$ and $f : A \cup X \longrightarrow \mathcal{N}$ by: $f(\alpha) = u$, $f(\beta) = m$, $f(\gamma) = n$, $f(\delta) = s$, $f(\mu) = n$, $f(q) = Q$, $f(p) = P$, $f(p') = P$, $f(r) = R$, $\mathcal{E} = \langle X, A, e \rangle$ where $e : X \longrightarrow \mathcal{P}(X \cup A)$ is defined by:

$$
\begin{array}{lcl}
e_\alpha = \{e_\beta, e_\gamma\} & \Longrightarrow & \left\{ \begin{array}{l} e_\beta \in e_\alpha \\ e_\gamma \in e_\alpha \end{array} \right. \Longrightarrow \left\{ \begin{array}{ll} \beta \mathbf{in} \alpha & \text{is true} \\ \gamma \mathbf{in} \alpha & \text{is true} \end{array} \right. \\
e_\beta = \{e_\delta, q\} & \Longrightarrow & \left\{ \begin{array}{l} e_\delta \in e_\beta \\ q \in e_\beta \end{array} \right. \Longrightarrow \left\{ \begin{array}{ll} \delta \mathbf{in} \beta & \text{is true} \\ q \mathbf{in} \beta & \text{is true} \end{array} \right. \\
e_\gamma = \{p\} & \Longrightarrow & \{ \ p \in e_\gamma \Longrightarrow \{ \ p \mathbf{in} \gamma \ \text{ is true} \\
e_\delta = \{e_\mu\} & \Longrightarrow & \{ \ e_\mu \in e_\delta \Longrightarrow \{ \ \mu \mathbf{in} \delta \ \text{ is true} \\
e_\mu = \{p', r\} & \Longrightarrow & \left\{ \begin{array}{l} p' \in e_\mu \\ r \in e_\mu \end{array} \right. \Longrightarrow \left\{ \begin{array}{ll} p' \mathbf{in} \mu & \text{is true} \\ r \mathbf{in} \mu & \text{is true} \end{array} \right.
\end{array}
$$

Hence $\mathcal{E}$ is described by the list of true atomical propositions. We construct the first matrix, $\mathbb{T}_1$, by setting the entry of column x and row y to 1, if the proposition $x \mathbf{in} y$ is true. All the empty entries are set to 0. See *Example table1.*

The second matrix, $\mathbb{T}_2$, has the same rows as $\mathbb{T}_1$ and as many columns as the number of prefixes forming the largest sequential chain in the process plus two. Actually, we have the first column, indexed by f, reporting the value of f applied to the row index. The remaining columns define F. The last column is filled by ε. In our example we have
$f(\alpha) = u$, $F(\alpha) = \langle \varepsilon, \varepsilon, ... \rangle$, $f(\beta) = m$, $F(\beta) = \langle \varepsilon, \varepsilon, ... \rangle$,
$f(\gamma) = n$, $F(\gamma) = \langle \varepsilon, \varepsilon, ... \rangle$, $f(\delta) = s$, $F(\delta) = \langle \varepsilon, \varepsilon, ... \rangle$,
$f(\mu) = n$, $F(\mu) = \langle out\ m, in\ m, \varepsilon \rangle$, $f(q) = Q$, $F(q) = \langle open\ n, \varepsilon \rangle$,
$f(p) = P$, $F(p) = \langle \varepsilon, \varepsilon, ... \rangle$, $f(p') = P$, $F(p') = \langle out\ s, open\ s, \varepsilon \rangle$,
$f(r) = R$, $F(r) = \langle \varepsilon, \varepsilon, ... \rangle$.
See *Example table2* for the construction of the matrix $\mathbb{T}_2$.

Example table1

$\mathbb{T}_1$	α	β	γ	δ	μ	q	p	p'	r
α	0	1	1	0	0	0	0	0	0
β	0	0	0	1	0	1	0	0	0
γ	0	0	0	0	0	0	1	0	0
δ	0	0	0	0	1	0	0	0	0
μ	0	0	0	0	0	0	0	1	1
q	0	0	0	0	0	0	0	0	0
p	0	0	0	0	0	0	0	0	0
p'	0	0	0	0	0	0	0	0	0
r	0	0	0	0	0	0	0	0	0

Example table2

$\mathbb{T}_2$	f	F
α	u	ε ε ...
β	m	ε ε ...
γ	n	ε ε ...
δ	s	ε ε ...
μ	n	$out\ m$ $in\ m$ ε
q	Q	$open\ n$ ε ...
p	P	ε ...
p'	P	$out\ s$ $open\ s$ ε
r	R	ε ...

The two matrices $\mathbb{T}_1$ and $\mathbb{T}_2$ suffice to describe each state S. Now we define the algorithm to compute the evolution of states under reductions.

Assume that the initial state S_1 is described by the tables $\mathbb{T}_1$ and $\mathbb{T}_2$. The algorithm will start by choosing randomly a row from table $\mathbb{T}_2$ such that its first position of the F-part is a prefix c with $c \neq \varepsilon$ and $c \neq input$[3].

If $c \neq \varepsilon$ and $c \neq input$ then we will proceed to *c-condition* where we check whether the condition depending on the entry c is true or not. If the condition is true we proceed with *c-reduction* that is the part of the algorithm which will update $\mathbb{T}_1$ and $\mathbb{T}_2$ to describe the next state S_2. We will have different *c-condition* and *c-reduction* for each type of prefix.

Denoting by $S_1 \models_{alg} S_2$ that S_2 is obtained from S_1, in one step, using our algorithm 4.1 instantiated with suitable $c - condition$ and $c - reduction$ functions, we can define the accessibility relation between states as:

$$S_1 \mathfrak{R} S_2 \text{ iff } S_1 \models_{alg} S_2$$

[3]This is a technical trick to avoid the choice of *input* and *output* prefixes in the same step

Algorithm 4.1 (General form of the accessibility algorithm).

begin
go:=true
while go do
    begin
    $Row = \{c \mid c \text{ is on the first row of } F \text{ part of } \mathbb{T}_2\}$
    while $Row \neq \emptyset$ *do*
        begin
        choose $c \in Row$
        if $c \neq \varepsilon$ *and* $c \neq input\ x$ *then*
            begin
            c-condition
            if condition then
                begin
                c-reduction*;*
                go:=false;
                end;
            else $Row := Row \setminus \{c\}$
            end;
        else $Row := Row \setminus \{c\}$
        end;
        go:=false;
    end;
end.

In the next subsections we define, for any reduction rule of the ambient calculus the corresponding $c - condition$ and $c - reduction$ functions.

4.1 *In*-Reduction

$$\text{(Red In): } n[in\ m.P|Q]|m[R]] \rightarrow [m[n[P|Q]|R]]$$

We wrap the two sides of the rule within the master ambient u:

$$u[n[in\ m.P|Q]|m[R]]] \rightarrow u[[m[n[P|Q]|R]]]$$

The state-process for the initial state is

$X = \{\alpha, \beta, \gamma\}$, $A = \{p, q, r\} \subseteq \mathfrak{V}$, $f_{S_1} : X \cup A \longrightarrow \mathcal{N}$, $f_{S_1}(\alpha) = u$, $f_{S_1}(\beta) = m$, $f_{S_1}(\gamma) = n$, $f_{S_1}(p) = P$, $f_{S_1}(q) = Q$, $f_{S_1}(r) = R$ and $F_{S_1} : X \cup A \longrightarrow \mathfrak{Str}$, $F_{S_1}(p) = \langle in\ m, c_1, c_2, ...\rangle$.

Informally, for the source state we have

$$S_1\text{: } \alpha\ [\ \beta\ [\ in\ m.p \mid q\ ]\ |\ \gamma\ [\ r\ ]\ ]$$

with the equations system

$$\begin{array}{lcl}
e_\alpha = \{e_\beta, e_\gamma\} & \Longrightarrow & \left\{ \begin{array}{l} e_\beta \in e_\alpha \\ e_\gamma \in e_\alpha \end{array} \right. \Longrightarrow \left\{ \begin{array}{l} \beta \mathbf{in} \alpha \text{ is true} \\ \gamma \mathbf{in} \alpha \text{ is true} \end{array} \right. \\
e_\beta = \{p, q\} & \Longrightarrow & \left\{ \begin{array}{l} p \in e_\beta \\ q \in e_\beta \end{array} \right. \Longrightarrow \left\{ \begin{array}{l} p \mathbf{in} \beta \quad \text{is true} \\ q \mathbf{in} \beta \quad \text{is true} \end{array} \right. \\
e_\gamma = \{r\} & \Longrightarrow & \{ \; r \in e_\gamma \Longrightarrow \{ \; r \mathbf{in} \gamma \text{ is true}
\end{array}$$

and matrices

Initial-In-table1

$\mathbb{T}_1$	α	β	γ	p	q	r
α	0	1	1	0	0	0
β	0	0	0	1	1	0
γ	0	0	0	0	0	1
p	0	0	0	0	0	0
q	0	0	0	0	0	0
r	0	0	0	0	0	0

Initial-In-table2

$\mathbb{T}_2$	f_{S_1}	F_{S_1}
α	u	
β	m	
γ	n	
p	P	$in\ m$, c_1, c_2,...
q	Q	
r	R	

Informally, for the target state, we have:

$$S_2\text{: } \alpha \; [\; \beta \; [\; \gamma \; [\; p \mid q \;] \mid r \;] \;]$$

with the equations system:

$$\begin{array}{lcl}
e_\alpha = \{e_\gamma\} & \Longrightarrow & \{ \; e_\gamma \in e_\alpha \Longrightarrow \{ \; \gamma \mathbf{in} \alpha \quad \text{is true} \\
e_\gamma = \{e_\beta, r\} & \Longrightarrow & \left\{ \begin{array}{l} e_\beta \in e_\gamma \\ r \in e_\gamma \end{array} \right. \Longrightarrow \left\{ \begin{array}{l} \beta \mathbf{in} \gamma \quad \text{is true} \\ r \mathbf{in} \gamma \quad \text{is true} \end{array} \right. \\
e_\beta = \{p, q\} & \Longrightarrow & \left\{ \begin{array}{l} p \in e_\beta \\ q \in e_\beta \end{array} \right. \Longrightarrow \left\{ \begin{array}{l} p \mathbf{in} \beta \quad \text{is true} \\ q \mathbf{in} \beta \quad \text{is true} \end{array} \right.
\end{array}$$

and the two matrices:

Final-In-table1

$\mathbb{T}_1$	α	β	γ	p	q	r
α	0	0	1	0	0	0
β	0	0	0	1	1	0
γ	0	1	0	0	0	1
p	0	0	0	0	0	0
q	0	0	0	0	0	0
r	0	0	0	0	0	0

Final-In-table2

$\mathbb{T}_2$	f_{S_1}	F_{S_1}
α	u	
β	m	
γ	n	
p	P	c_1, c_2,...
q	Q	
r	R	

Therefore, the only changes are:

$$\begin{array}{lll|lll}
S_1\text{:} & \beta \mathbf{in} \gamma \; true & & S_2\text{:} & \beta \mathbf{in} \gamma \; false \\
 & \beta \mathbf{in} \gamma \; false & \Longrightarrow & & \beta \mathbf{in} \gamma \; true
\end{array}$$

$F_{S_2}(p) = F_{S_2}(p) - \langle in\ m \rangle$.

Further we develop the algorithm 4.2 for in-condition. First it computes $f_{S_1}^{-1}(m)$ to check whether it can apply $in\ m$. If $f_{S_1}^{-1}(m)$ has many elements

it will choose randomly one: $\nu \in f_{S_1}^{-1}(m)$ and it will test the correct nesting of ambients to apply $(Red - In)$. If the condition is not satisfied, then the algorithm will choose another element and will continue in this way until the correct nesting is satisfied for one element, if any, if not it will go further. Then it will proceed to *in-reduction* to update $\mathbb{T}_1$ and $\mathbb{T}_2$ for S_2.

Algorithm 4.2 (In-condition algorithm).

begin
condition:=true;
go:=true;
while go do
  begin
  if $f_{S_1}^{-1}(m) \neq \emptyset$ *then*
    choose $\nu \in f_{S_1}^{-1}(m)$
    if parent(parent(p))=parent(ν*) then*
      go:=false
    else $f_{S_1}^{-1}(m) := f_{S_1}^{-1}(m) \backslash \{\nu\}$
  else
    begin
    condition:=false;
    go:=false;
    end;
  end;
end.

Some remarks: $f_{S_1}^{-1}(m)$ can be computed by checking in the column of f_{S_1}, of table $\mathbb{T}_2$, where exactly m appears and by identifying the urelements for which it is appearing. *parent(*α*)* can be found, in the matrix $\mathbb{T}_1$, by checking on the column of α where 1 appears[4]. We can read, in the row where 1 appears, which is the parent of α.

Algorithm 4.3 (In-reduction algorithm).

begin
update $\mathbb{T}_2$*:* $F_{S_2}(p) := F_{S_1}(p) - \langle in\ m \rangle$
$F_{S_2}(x) = F_{S_1}(x)$ *for* $x \neq p$
$f_{S_2}(x) := f_{S_1}(x)$
update $\mathbb{T}_1$*:* $\beta \mathfrak{in} \alpha := 0$ *(i.e.* $\text{parent}(p) \mathfrak{in} \text{parent}(\text{parent}(p))$ *becomes false)*
$\beta \mathfrak{in} \gamma := 1$ *(*$\text{parent}(p) \mathfrak{in} \nu$ *becomes true)*
end.

4.2 Out-Reduction

$$(Red\ Out) : m\,[n\,[out\ m.P \mid Q] \mid R] \longrightarrow m\,[R] \mid n\,[P \mid Q]$$

[4]if α is not the master ambient, it appears once

We wrap the two sides of the rule within the master ambient u: $u\,[m\,[n\,[out\ m.P \mid Q]\ \mid R]] \longrightarrow u\,[m\,[R]\ \mid n\,[P \mid Q]]$. The state-process for the initial state is:

$X = \{\alpha, \beta, \gamma\}$, $A = \{p, q, r\} \subseteq \mho$, $f_{S_1} : X \cup A \longrightarrow \mathcal{N}$, $f_{S_1}(\alpha) = u$, $f_{S_1}(\beta) = m$, $f_{S_1}(\gamma) = n$, $f_{S_1}(p) = P$, $f_{S_1}(q) = Q$, $f_{S_1}(r) = R$ and $F_{S_1} : X \cup A \longrightarrow \mathfrak{Stx}$, $F_{S_1}(p) = \langle out\ m, c_1, c_2, ...\rangle$.

Informally we have:

S_1: $\alpha\,[\beta\,[\gamma\,[out\ m.p \mid q]\ \mid r]]$ with the equations system

$$\begin{array}{lcl}
e_\alpha = \{\beta\} & \Longrightarrow & \{\ e_\beta \in e_\alpha \Longrightarrow \{\ \beta \mathbf{in} \alpha \quad \text{is true} \\
e_\beta = \{e_\gamma, r\} & \Longrightarrow & \left\{\begin{array}{l} e_\gamma \in e_\beta \\ r \in e_\beta \end{array}\right. \Longrightarrow \left\{\begin{array}{ll} \gamma \mathbf{in} \beta & \text{is true} \\ r \mathbf{in} \beta & \text{is true} \end{array}\right. \\
e_\gamma = \{p, q\} & \Longrightarrow & \left\{\begin{array}{l} p \in e_\gamma \\ q \in e_\gamma \end{array}\right. \Longrightarrow \left\{\begin{array}{ll} p \mathbf{in} \gamma & \text{is true} \\ q \mathbf{in} \gamma & \text{is true} \end{array}\right.
\end{array}$$

and matrices

Initial-Out-table1

$\mathbb{T}_1$	α	β	γ	p	q	r
α	0	1	0	0	0	0
β	0	0	1	0	0	1
γ	0	0	0	1	1	0
p	0	0	0	0	0	0
q	0	0	0	0	0	0
r	0	0	0	0	0	0

Initial-Out-table2

$\mathbb{T}_2$	f_{S_1}	F_{S_1}
α	u	
β	m	
γ	n	
p	P	$out\ m$, c_1, c_2,...
q	Q	
r	R	

S_2: $\alpha\,[\beta\,[r]\ \mid \gamma\,[p \mid q]]$ with the equations system

$$\begin{array}{lcl}
e_\alpha = \{e_\beta, e_\gamma\} & \Longrightarrow & \left\{\begin{array}{l} e_\beta \in e_\alpha \\ e_\gamma \in e_\alpha \end{array}\right. \Longrightarrow \left\{\begin{array}{ll} \beta \mathbf{in} \alpha & \text{is true} \\ \gamma \mathbf{in} \alpha & \text{is true} \end{array}\right. \\
e_\beta = \{r\} & \Longrightarrow & \{\ r \in e_\beta \Longrightarrow \{\ r \mathbf{in} \beta \quad \text{is true} \\
e_\gamma = \{p, q\} & \Longrightarrow & \left\{\begin{array}{l} p \in e_\gamma \\ q \in e_\gamma \end{array}\right. \Longrightarrow \left\{\begin{array}{ll} p \mathbf{in} \gamma & \text{is true} \\ q \mathbf{in} \gamma & \text{is true} \end{array}\right.
\end{array}$$

and matrices

Final-Out-table1

$\mathbb{T}_1$	α	β	γ	p	q	r
α	0	1	1	0	0	0
β	0	0	0	0	0	1
γ	0	0	0	1	1	0
p	0	0	0	0	0	0
q	0	0	0	0	0	0
r	0	0	0	0	0	0

Final-Out-table2

$\mathbb{T}_2$	f_{S_1}	F_{S_1}
α	u	
β	m	
γ	n	
p	P	c_1, c_2,...
q	Q	
r	R	

Therefore, the only changes are:

$$\begin{array}{rl|rl}
S_1: & \gamma \mathbf{in} \alpha\ true & \Longrightarrow & S_2: \quad \gamma \mathbf{in} \alpha\ false \\
& \gamma \mathbf{in} \beta\ false & & \gamma \mathbf{in} \beta\ true
\end{array}$$

$F_{S_2}(p) = F_{S_2}(p) - \langle out\ m\rangle$.

Algorithm 4.4 (*Out*-condition algorithm).

begin
condition:=true;
go:=true;
while go do
  begin
  if $f_{S_1}^{-1}(m) \neq \emptyset$ *then*
    choose $\nu \in f_{S_1}^{-1}(m)$
    if parent(parent(p))=ν *then*
      go:=false
    else $f_{S_1}^{-1}(m) := f_{S_1}^{-1}(m) \backslash \{\nu\}$
  else
    begin
    condition:=false;
    go:=false;
    end;
  end;
end.

Algorithm 4.5 (*Out*-reduction algorithm).

begin
update $\mathbb{T}_2$: $F_{S_2}(p) := F_{S_1}(p) - \langle out\ m \rangle$;
$F_{S_2}(x) = F_{S_1}(x)$ *for all* $x \neq p$;
$f_{S_2}(x) := f_{S_1}(x)$;
update $\mathbb{T}_1$: $\gamma \mathbf{in} \alpha := 0$
$\gamma \mathbf{in} \beta$:= *1*
end.

4.3 *Open*-Reduction

$$(Red\ Open) : open\ n.P \mid n\,[Q] \longrightarrow P \mid Q$$

We wrap the two sides of the rule within the master ambient u:
$u\,[open\ n.P \mid n\,[Q]] \longrightarrow u\,[P \mid Q]$
$X = \{\alpha, \beta\}$, $A = \{p, q\} \subseteq \mathfrak{V}$, $f_{S_1} : X \cup A \longrightarrow \mathcal{N}$, $f_{S_1}(\alpha) = u$, $f_{S_1}(\beta) = n$, $f_{S_1}(p) = P$, $f_{S_1}(q) = Q$, $F_{S_1}(p) = \langle open\ n, c_1, c_2, ... \rangle$.
S_1: $\alpha\,[open\ n.p \mid \beta\,[q]]$

$$\begin{array}{rcl} e_\alpha = \{p, e_\beta\} & \Longrightarrow & \left\{ \begin{array}{l} p \in e_\alpha \\ e_\beta \in e_\alpha \end{array} \right. \Longrightarrow \left\{ \begin{array}{ll} q \mathbf{in} \alpha & \text{is } true \\ \beta \mathbf{in} \alpha & \text{is } true \end{array} \right. \\ e_\beta = \{q\} & \Longrightarrow & \{\ q \in e_\beta \quad \Longrightarrow \{\ q \mathbf{in} \beta \quad \text{is } true \end{array}$$

S_2: $\alpha\,[p \mid q]$

$$e_\alpha = \{p, q\} \quad \Longrightarrow \quad \left\{ \begin{array}{l} p \in e_\alpha \\ q \in e_\alpha \end{array} \right. \Longrightarrow \left\{ \begin{array}{ll} p \mathbf{in} \alpha & \text{is } true \\ q \mathbf{in} \alpha & \text{is } true \end{array} \right.$$

The changes:

$$\begin{array}{rl|rl} S_1: & \beta \mathbf{in} \alpha\ true & & \beta \mathbf{in} \alpha\ false \\ & q \mathbf{in} \alpha\ false & \Longrightarrow & q \mathbf{in} \alpha\ true \\ & q \mathbf{in} \beta\ true & & q \mathbf{in} \beta\ false \end{array}$$

$F_{S_2}(p) = F_{S_2}(p) - \langle open\ n \rangle$.

Initial-Open-table1

$\mathbb{T}_1$	α	β	p	q
α	0	1	1	0
β	0	0	0	1
p	0	0	0	0
q	0	0	0	0

Initial-Open-table2

$\mathbb{T}_2$	f_{S_1}	F_{S_1}
α	u	
β	n	
p	P	*open* n, c_1, c_2,...
q	Q	

Final-Open-table1

$\mathbb{T}_1$	α	β	p	q
α	0	0	1	1
β	0	0	0	0
p	0	0	0	0
q	0	0	0	0

Final-Open-table2

$\mathbb{T}_2$	f_{S_1}	F_{S_1}
α	u	
β	n	
p	P	c_1, c_2,...
q	Q	

In spite of the fact that, in S_2, the ambient n, alias β, disappears we keep it in our tables. This is important for the situations in which many urelements are associated with n, i.e. n appears in many places. But the fact that β is still in our matrices does not change S_2 at all. If we look to $\mathbb{T}_1$ for S_1, not the column, nor the row of β contains 1, so β is not parent and not child, not element of a set and does not contain elements, i.e. does not exist!

Algorithm 4.6 (*Open*-condition algorithm).

begin
condition:=true;
go:=true;
while go do
  begin
  if $f_{S_1}^{-1}(n) \neq \emptyset$ *then*
    choose $\nu \in f_{S_1}^{-1}(m)$
    if parent(p)=parent(ν*) then*
      go:=false
    else $f_{S_1}^{-1}(m) := f_{S_1}^{-1}(m) \backslash \{\nu\}$
  else
    begin
    condition:=false;
    go:=false;
    end;
  end;
end.

Algorithm 4.7 (*Open*-reduction algorithm).

begin	
update $\mathbb{T}_2$*:*	$F_{S_2}(p) := F_{S_1}(p) - \langle open\ m \rangle$,
	$F_{S_2}(x) = F_{S_1}(x)$ *for* $x \neq p$
	$f_{S_2}(x) := f_{S_1}(x)$
update $\mathbb{T}_1$*:*	$\beta \mathfrak{in} \alpha := 0$
	$q \mathfrak{in} \alpha := 1$
	$q \mathfrak{in} \beta := 0$
end.	

4.4 *Communication*-reduction

We distinguish two main cases:

1. when the name of an ambient is sent: $\langle n \rangle \mid (x).x\,[P] \mid R \longrightarrow n\,[P] \mid R$
2. when the name sent is part of a capability $\langle n \rangle \mid (x)\ in\ x.P \mid R \longrightarrow in\ n.P \mid R$

We will treat separately each case, because in the first one we have to chose an urelement for x while in the second not. This will determine different changes in $\mathbb{T}_1$. In the end we will have only one algorithm for communication that will solve both cases. Moreover, if will imagine the case of combining the two situations, our algorithm will compute properly the target state.

$$\text{Case1: } \langle n \rangle \mid (x).x\,[P] \mid R \longrightarrow n\,[P] \mid R$$

We treat x as any other ambient name.

We wrap both states in the master ambient: $u\,[\langle n \rangle |\ (x).x\,[P] \mid R] \longrightarrow u\,[n\,[P] \mid R]$
$X = \{\alpha, \beta, \kappa\}$, $A = \{p, r\}$, $f_{S_1} : A \cup X \longrightarrow \mathcal{N}$, $f_{S_1}(\alpha) = u$, $f_{S_1}(\beta) = n$, $f_{S_1}(\kappa) = x$, $f_{S_1}(p) = P$, $f_{S_1}(q) = Q$, $F_{S_1} : X \cup A \longrightarrow \mathfrak{Str}$, $F_{S_1}(\beta) = \langle output, \varepsilon, \varepsilon, ... \rangle$, $F_{S_1}(\kappa) = \langle input\ x, c_1, c_2, ... \rangle$

S_1: $\alpha\,[output.\beta\,[\ ] \mid input\ x.\kappa\,[\ p\ ] \mid r]$

$$e_\alpha = \{e_\beta, e_\kappa, r\} \implies \begin{cases} e_\beta \in e_\alpha \\ e_\kappa \in e_\alpha \\ r \in e_\alpha \end{cases} \implies \begin{cases} \beta \mathfrak{in} \alpha & \text{is } true \\ \kappa \mathfrak{in} \alpha & \text{is } true \\ r \mathfrak{in} \alpha & \text{is } true \end{cases}$$

$$e_\beta = \emptyset$$

$$e_\kappa = \{p\} \implies \{\ p \in e_\kappa \implies \{\ p \mathfrak{in} \kappa \quad \text{is } true$$

Initial-Comm1-table1

$\mathbb{T}_1$	α	β	κ	p	r
α	0	1	1	0	1
β	0	0	0	0	0
κ	0	0	0	1	0
p	0	0	0	0	0
r	0	0	0	0	0

Initial-Comm1-table2

$\mathbb{T}_2$	f_{S_1}	F_{S_1}
α	u	
β	n	$output, \varepsilon, \varepsilon, ...$
κ	x	$input\ x, c_1, c_2, ...$
p	P	
r	R	

S_2: $\alpha\,[\beta\,[\ p\ ]\ |\ r\ ]$

$$e_\alpha = \{e_\beta, r\} \implies \begin{cases} e_\beta \in e_\alpha \\ r \in e_\alpha \end{cases} \implies \begin{cases} \beta \text{in} \alpha & \text{is } true \\ r \text{in} \alpha & \text{is } true \end{cases}$$
$$e_\beta = \{p\} \implies \{\ p \in e_\beta \implies \{\ p \text{in} \beta \quad \text{is } true$$

Final-Comm1-table1

$\mathbb{T}_1$	α	β	κ	p	r
α	0	1	0	0	1
β	0	0	0	1	0
κ	0	0	0	0	0
p	0	0	0	0	0
r	0	0	0	0	0

Final-Comm1-table2

$\mathbb{T}_2$	f_{S_1}	F_{S_1}
α	u	
β	n	$c'_1, c'_2, ...$
κ	x	$\varepsilon, \varepsilon, ...$
p	P	
r	R	

The changes:

$$\begin{array}{rl|rl} S_1: & \kappa \text{in} \alpha\ true & & S_2: \quad \kappa \text{in} \alpha\ false \\ & p \text{in} \beta\ false & \implies & \quad\ p \text{in} \beta\ true \\ & p \text{in} \kappa\ true & & \quad\ p \text{in} \kappa\ false \end{array}$$

$F_{S_2}(\kappa) = \langle \varepsilon, ... \rangle$.

$F_{S_2}(\beta) = (F_{S_1}(\kappa) - \langle input\ x \rangle)\,(x \leftarrow f_{S_1}(\beta))$ where $f\,(x)\,(x \leftarrow y)$ denotes the substitution of x by y in all occurrences of x in $f(x)$. In our case, in the string $F_{S_1}(\kappa) - \langle input\ x \rangle$ all occurrences of x are replaced by $n = f_{S_1}(\beta)$.

Before writing the algorithm, we will analyze the other kind of communication also:

$$\text{Case 2: } (x).in\ x.P \mid \langle n \rangle \mid R \longrightarrow in\ n.P \mid R$$

first step: $u\,[(x).in\ x.P \mid \langle n \rangle \mid R] \longrightarrow u\,[in\ n.P \mid R]$

$X = \{\alpha, \beta\}$, $A = \{p, r\} \subseteq \mho$, $f_{S_1} : X \cup A \longrightarrow \mathcal{N}$, $f_{S_1}(\alpha) = u$, $f_{S_1}(\beta) = n$, $f_{S_1}(p) = P$, $f_{S_1}(r) = R$, $F_{S_1}(\beta) = \langle output, \varepsilon, ... \rangle$, $F_{S_1}(p) = \langle input\ x, in\ x, c_1, c_2, ... \rangle$

Because x is not an ambient, nor an atomical process in our formula, we will not choose an urelement for it.

S_1: $\alpha\,[input\ x.in\ x.p \mid output.\beta\,[\ ]\ |\ r\ ]$

$$e_\alpha = \{p, e_\beta, r\} \implies \begin{cases} p \in e_\alpha \\ e_\beta \in e_\alpha \\ r \in e_\alpha \end{cases} \implies \begin{cases} p \text{in} \alpha & \text{is } true \\ \beta \text{in} \alpha & \text{is } true \\ r \text{in} \alpha & \text{is } true \end{cases}$$
$$e_\beta = \emptyset$$

Initial-Comm2-table1

$\mathbb{T}_1$	α	β	p	r
α	0	1	1	1
β	0	0	0	0
p	0	0	0	0
r	0	0	0	0

Initial-Comm2-table2

$\mathbb{T}_2$	f_{S_1}	F_{S_1}
α	u	
β	n	$output, \varepsilon, \varepsilon, ...$
p	P	$input\ x, in\ x, c_1, c_2, ...$
r	R	

S_2: $\alpha\,[in\ n.p \,|\, r\ ]$

$$e_\alpha = \{p, r\} \implies \begin{cases} p \in e_\alpha \\ r \in e_\alpha \end{cases} \implies \begin{cases} p\mathbf{in}\alpha & \text{is } true \\ r\mathbf{in}\alpha & \text{is } true \end{cases}$$

Final-Comm2-table1

$\mathbb{T}_1$	α	β	p	r
α	0	0	1	1
β	0	0	0	0
p	0	0	0	0
r	0	0	0	0

Final-Comm2-table2

$\mathbb{T}_2$	f_{S_1}	F_{S_1}
α	u	
β	n	$\varepsilon, \varepsilon, ...$
p	P	$in\ x, c'_1, c'_2, ...$
r	R	

The changes:

S_1: $\beta\mathbf{in}\alpha\ true \longrightarrow S_2$: $\beta\mathbf{in}\alpha\ false$

$F_{S_2}(\beta) = F_{S_1}(\beta) - \langle output \rangle$, $F_{S_2}(p) = (F_{S_1}(p) - \langle input\ x \rangle)\,(x \leftarrow f(\beta))$

Now we will construct the algorithms. It is possible to treat both cases of communication together.

Algorithm 4.8 (*communication*-condition).

```
begin
condition:=true;
go:=true;
Input := {ν ∈ X ∪ A | F_S1(ν) = ⟨input x, ...⟩}
while go do
      begin
      if Input ≠ ∅ then
            choose ν ∈ Input
            if parent(ν)=parent(β) then
                  go:=false
            else Input := Input\{ν}
      else
            begin
            condition:=false;
            go:=false;
            end;
      end;
end.
```

Algorithm 4.9 (communication-reduction algorithm).

begin
if $f(\nu) = x$ *then (we are in the first case)*
  update $\mathbb{T}_2$*:* $F_{S_2}(\beta) := (F_{S_1}(\kappa) - \langle input\ x\rangle)\,(x \leftarrow f_{S_1}(\beta))$
  $F_{S_2}(\kappa) := \langle \varepsilon, \varepsilon, ...\rangle$
  $f_{S_2}(x) := f_{S_1}(x)$
  update $\mathbb{T}_1$*:* $x\mathbf{in}\alpha := 0$
  $p\mathbf{in}\beta := 1$
  $p\mathbf{in}\kappa := 0$
else (we are in the second case)
  update $\mathbb{T}_2$*:* $F_{S_2}(\beta) := F_{S_1}(\beta) - \langle output\rangle$
  $F_{S_2}(p) := (F_{S_1}(p) - \langle input\ x\rangle)\,(x \leftarrow f_{S_1}(\beta))$
  $f_{S_2}(x) := f_{S_1}(x)$
  update $\mathbb{T}_1$*:* $\beta\mathbf{in}\alpha := 0$
end.

5 Conclusions

Our approach to Ambient Calculus opens the perspective of using model checking algorithms (or software) developed for temporal logics for analyzing mobile computations. This is because we found a way of implementing the information behind the ambient processes, in the form of state-processes described by two matrices, and we constructed the algorithms to calculate the accessibility relation between states. Moreover, our algorithms have no problems from the complexity point of view.

Having the description of the states, together with the algorithms for accessibility relation, all we have to do for having model checking for mobile computations, is to use further the algorithms for model checking CTL* (a CTL is possible also, see [6]) and we intend to study further this possibility.

References

[1] J. Barwise and L. Moss. *Vicious Circles. On the Mathematics of Non-Wellfounded Phenomena*. CLSI Lecture Notes Number 60 Stanford: CSLI Publication, 1996.

[2] L. Cardelli and A.D. Gordon. Ambient logic. *http://www.luca.demon.co.uk/*.

[3] L. Cardelli and A.D. Gordon. Anytime, anywhere. modal logics for mobile ambients. pages 365–377, 2000.

[4] L. Cardelli and A.D. Gordon. Mobile ambients. *Theoretical Computer Science, Special Issue on Coordination, D. Le Mtayer Editor*, pages 177–213, June 2000.

[5] E. A. Emerson. Temporal and modal logic. *Handbook of Theoretical Computer Science*, B: Formal Models and Sematics:995–1072, 1990.

[6] R. Mardare and C. Priami. A propositional branching temporal logic for ambient calculus. Technical report, Dipartimento di Informatica e Tlc, University of Trento, 2003. Available at www.dit.unitn.it following the link Publications.